SMEC:Rethinking Matryoshka Representation Learning for Retrieval Embedding Compression

Biao Zhang, Lixin Chen, Tong Liu
Taobao & Tmall Group of Alibaba
Hangzhou, China
{zb372670,tianyou.clx,yingmu}@taobao.com

Bo Zheng
Taobao & Tmall Group of Alibaba
Beijing, China
bozheng@alibaba-inc.com

Abstract

Large language models (LLMs) generate high-dimensional embeddings that capture rich semantic and syntactic information. However, high-dimensional embeddings exacerbate computational complexity and storage requirements, thereby hindering practical deployment. To address these challenges, we propose a novel training framework named Sequential Matryoshka Embedding Compression (SMEC). This framework introduces the Sequential Matryoshka Representation Learning(SMRL) method to mitigate gradient variance during training, the Adaptive Dimension Selection (ADS) module to reduce information degradation during dimension pruning, and the Selectable Cross-batch Memory (S-XBM) module to enhance unsupervised learning between high- and low-dimensional embeddings. Experiments on image, text, and multimodal datasets demonstrate that SMEC achieves significant dimensionality reduction while maintaining performance. For instance, on the BEIR dataset, our approach improves the performance of compressed LLM2Vec embeddings (256 dimensions) by 1.1 points and 2.7 points compared to the Matryoshka-Adaptor and Search-Adaptor models, respectively.

1 Introduction

Large language models excel in diverse text tasks due to their ability to capture nuanced linguistic structures and contextual dependencies. For instance, GPT-4 achieves state-of-the-art performance on benchmarks like GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019), demonstrating its proficiency in tasks such as natural language inference (NLI), question answering (QA), and text classification. This success is attributed to their transformer-based architectures (Vaswani et al., 2017), which enable parallel processing of sequential data and capture long-range dependencies through self-attention mechanisms. Similarly, Llama-3 (Grattafiori et al., 2024) and ChatGPT (Brown et al., 2020) leverage similar principles to achieve comparable or superior performance in domain-specific and multi-lingual tasks.

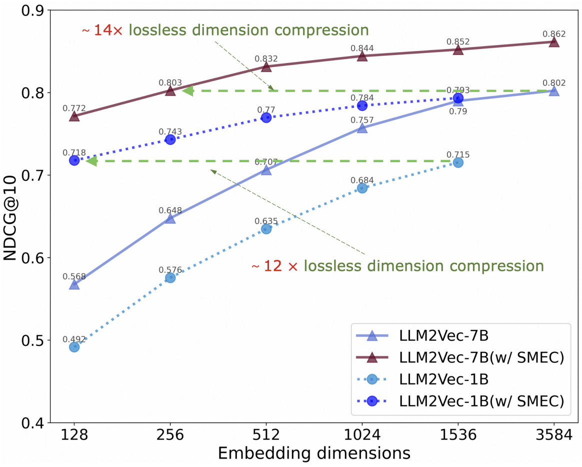
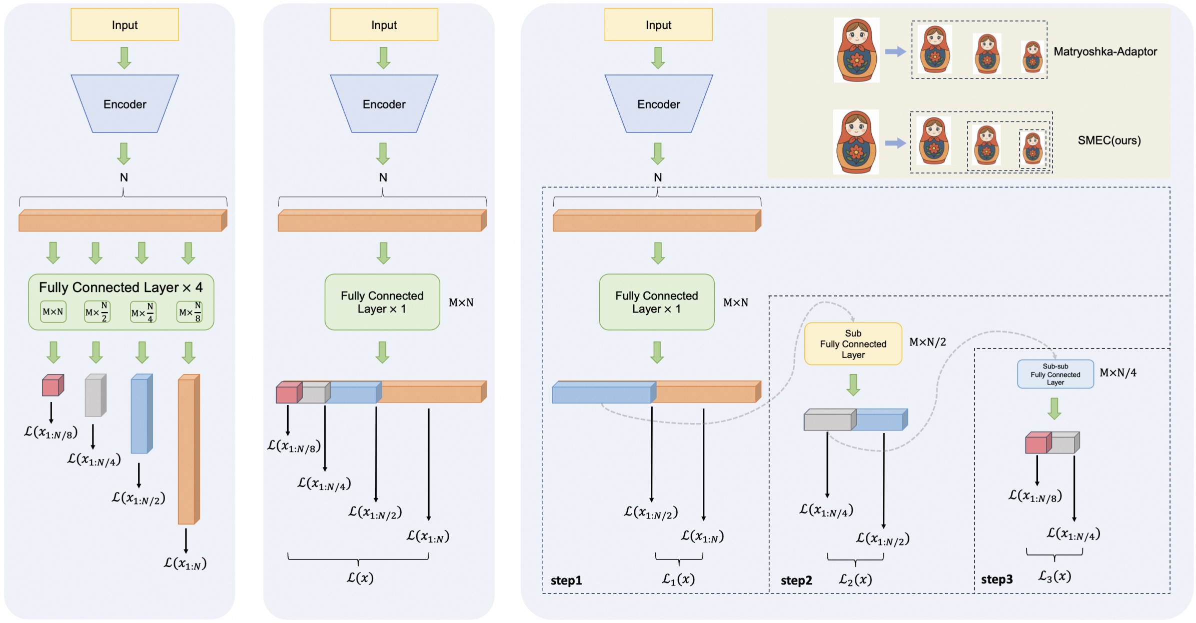


Figure 1: The effectiveness of the SMEC in dimensionality reduction. After customized training with the SMEC method on BEIR Quora dataset, the embeddings of LLM2Vec-7B (3584 dimensions) and LLM2Vec-1B (1536 dimensions) can achieve 14× and 12× lossless compression, respectively.

LLMs are increasingly integrated into commercial information retrieval (IR) systems, such as search engines (e.g., Google's MUM) and recommendation platforms (e.g., Netflix's content retrieval). Their ability to generate embeddings for long documents (e.g., books, research papers) and dynamic queries (e.g., conversational search) makes them indispensable for modern applications. For example, the BEIR benchmark (Thakur et al., 2021) evaluates cross-domain retrieval performance, where LLMs outperform traditional BM25(Robertson and Walker, 1994) and BERT-based models(Devlin et al., 2019) by leveraging contextual embeddings.

While LLMs' high-dimensional embeddings enable sophisticated semantic modeling, their storage

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 26209–26222
November 4-9, 2025 ©2025 Association for Computational Linguistics



(a) Search-Adaptor (b) Matryoshka-Adaptor (c) SMEC(ours)

Figure 2: Illustration of embedding compression architectures and our proposed approach. (a) presents the direct feature dimensionality reduction performed by the Search-Adaptor using FC layers. (b) illustrates the Matryoshka-Adaptor, which employs a shared set of FC layers to generate low-dimensional embeddings with multiple output dimensions. A Matryoshka-like hierarchical inclusion relationship exists between the high- and low-dimensional embeddings. (c) presents our proposed Sequential Matryoshka Embedding Compression (SMEC) framework, which adopts a sequential approach to progressively reduce high-dimensional embeddings to the target dimension. The animated diagram in the upper-right corner vividly highlights the distinction between Matryoshka-Adaptor and SMEC.

and computational costs hinder scalability. Embedding dimensions of LLMs typically range from 1,024 (e.g., GPT-3) to 4,096 (e.g., Llama-3), exacerbating storage overhead and computational inefficiency—especially in real-time systems requiring dynamic updates. Moreover, high-dimensional vectors degrade the performance of retrieval algorithms due to the curse of dimensionality (Beyer et al., 1999). For example: exact nearest-neighbor search in high-dimensional spaces becomes computationally infeasible, necessitating approximate methods like FAISS (Johnson et al., 2017) or HNSW (Yury et al., 2018). Even with optimizations, query latency increases exponentially with dimensionality, limiting responsiveness in real-world applications.

To address these challenges, Matryoshka Representation Learning (MRL) (Kusupati et al., 2022) encodes multi-scale information into a single embedding, balancing task complexity and efficiency. It achieves strong results in large-scale classification and retrieval tasks and has inspired variants like Matryoshka-Adaptor (Yoon et al., 2024), which offers a scalable framework for transforming embeddings into structured representations with Matryoshka properties under both supervised and unsupervised settings. However, MRL's multi-scale parallel training strategy simultaneously limits its practical application in industry. When the retrieval system requires a new low-dimensional embedding, retraining from scratch is necessary to achieve effective dimensionality reduction.

In this paper, we systematically analyze the limitations of MRL and its variants in embedding compression and propose three key enhancements: (1) a continued-training-friendly training framework named Sequential Matryoshka Representation Learning (SMRL); (2) an adaptive dimension selection (ADS) mechanism to minimize information degradation during dimension pruning; and (3) a Selectable Cross-batch Memory (S-XBM) strategy to enhance unsupervised learning between high- and low-dimensional embeddings.

2 Related Work

2.1 Matryoshka representation learning

Matryoshka representation learning introduces a novel paradigm where embeddings are pretrained to inherently support progressive dimension truncation. This enables fine-grained control over the trade-off between computational latency (via reduced dimensionality) and accuracy (via retained semantic structure). Key innovations include the design of Matryoshka properties, such as hierarchical information encoding and intra-cluster compactness, which ensure that even truncated embeddings retain utility for downstream tasks.

In addition to representation learning, the concept of MRL have been applied to image generation, such as Matryoshka Diffusion Models (MDM) (Gu et al., 2023); multimodal content understanding, such as M^3 (Cai et al., 2024); and Multimodal Large Language Model (MLLM), such as Matryoshka Query Transformer (MQT) (Hu et al., 2024).

2.2 Embedding Compression

Embedding compression aims to reduce the computational and memory footprint of neural network models or embeddings while preserving their utility for downstream tasks. This objective has driven research across multiple paradigms, each addressing different trade-offs between compression efficiency, performance retention, and adaptability. Early approaches primarily focused on unsupervised techniques based on linear algebra, such as Principal Component Analysis (PCA) (Jolliffe and Cadima, 2016), Linear Discriminant Analysis (LDA) (Mclachlan), and Non-negative Matrix Factorization (NMF) (Lee and Seung, 2000). Building upon these, autoencoders and their variants, such as Variational Autoencoders (VAEs) (Kingma et al., 2013), have gradually emerged as powerful tools for nonlinear dimensionality reduction, capable of capturing complex data distributions. With the development of deep learning, methods such as Contrastive Predictive Coding (CPC) (Oord et al., 2018) and Momentum Contrast (MoCo) (He et al., 2020) are capable of learning robust and compact representations from unlabeled data.

Recently, customized methods such as Search-Adaptor (Yoon et al., 2023) and Matryoshka-Adaptor (Yoon et al., 2024) have emerged as a new trend in embedding compression. They achieve significant dimensionality reduction by adding only a small number of parameters to the original representation model and retraining it on specific data.

3 Method

3.1 Rethinking MRL for embedding compression

MRL employs a nested-dimensional architecture to train models that learn hierarchical feature representations across multiple granularities. This allows adaptive deployment of models based on computational constraints. Specifically, MRL defines a series of models $f_1, f_2, \ldots, f_M$ that share identical input and output spaces but progressively expand their hidden dimensions.

The term Matryoshka derives from the hierarchical parameter structure where the parameters of model f_m are nested within those of its successor f_{m+1}. To illustrate, consider a FC layer within the largest model f_M, which contains d_M neurons in its hidden layer. Correspondingly, the FC layer of f_m retains the first d_m neurons of this structure, with dimensions satisfying $d_1 \leq d_2 \leq \cdots \leq d_M$. MRL jointly trains these models using the following objective:

$$\sum_{m=1}^{M} c_m \cdot \mathcal{L}(f_m(\mathbf{x}); y), \tag{1}$$

where $\mathcal{L}$ denotes the loss function, y represents the ground-truth label, and c_m are task-specific weighting coefficients. Notably, each training iteration requires forward and backward propagation for all M models, resulting in substantial computational overhead compared to training a single standalone model. Upon convergence, MRL enables flexible inference by selecting any intermediate dimension $d_i \leq d_M$, thereby accommodating diverse computational constraints.

Although the MRL method has partially mitigated the performance degradation of representations during dimensionality reduction, we contend that it still faces the following three unresolved issues:

Gradient Fluctuation. In large-scale vector retrieval systems, sample similarity is measured by the distance between their representation vectors. Consequently, the optimization of embedding models typically employs loss functions based on embedding similarity. In this condition, according to the derivation in Appendix A, the loss function $\mathcal{L}^d$ of MRL under dimension d satisfies the following relationship with respect to the parameter $\mathbf{w}_i$ in the i-th dimension of the FC layer:

$$\frac{\partial \mathcal{L}^d}{\partial \mathbf{w}_i} \propto \frac{1}{\delta(d)^2}. \tag{2}$$

Here, $\delta(d)$ is a complex function that is positively correlated with the dimension d. This equation provides a mathematical foundation for analyzing gradient fluctuations in multi-dimensional joint optimization architecture. It indicates that during the MRL training process, loss functions from various dimensions result in gradients of varying magnitudes on the same model parameter, thereby increasing gradient variance. In Section 5.2, we empirically demonstrated that the conclusion above is applicable to different loss functions. We propose a solution to resolve the aforementioned problem in Section 3.2.

Information Degradation. Neural network parameters exhibit heterogeneous contributions to model performance, as demonstrated by the non-uniform distribution of their gradients and feature importance metrics (Frankle and Carbin, 2018). The MRL method employs a dimension truncation strategy (e.g., $D \to D/2 \to D/4 \ldots$) to prune parameters and reduce feature dimensions by retaining partial parameters. However, this approach fails to adequately preserve critical parameters because it relies on a rigid, static truncation rule. Although MRL employs joint training of high- and low-dimensional vectors to redistribute information between truncated and retained parameters, this process is unavoidably accompanied by information degradation. Specifically, discarded parameters may contain essential information, such as unique feature mappings or high-order dependencies, that cannot be effectively recovered by the remaining ones. Empirical evidence, such as accuracy degradation and increased generalization gaps, demonstrates that such loss leads to suboptimal model performance and slower convergence (Li et al., 2023). In summary, while MRL enables hierarchical dimensionality reduction, its inability to selectively retain critical parameters and the inherent information degradation during post-truncation training ultimately undermine its effectiveness in maintaining model performance. In Section 3.3, we propose a more effective dimension pruning method.

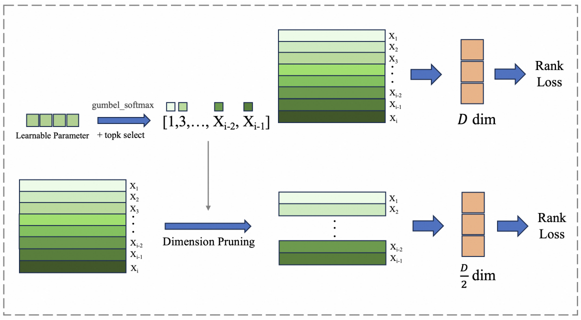


Figure 3: The ADS module introduces a set of learnable parameters to dynamically select dimensions based on their importance during the dimensionality reduction process.

Sample Selection. The MRL framework employs supervised learning to jointly train high-dimensional (D) and low-dimensional (D') features. However, the number of available samples is limited by manual annotation. Matryoshka-Adaptor introduces in-batch sample mining strategies to expand the training sample scale, thereby addressing the inherent limitation. Specifically, it generates cross-sample pairs via the cartesian product of batch samples:

$$\mathcal{P} = \{(x_i, x_j) \mid x_i, x_j \in \text{Batch},\ i \neq j\}. \tag{3}$$

This approach creates $B(B-1)$ pairs per batch (where B denotes the batch size), enabling cross-sample comparisons within large batches. However, this indiscriminate pairing introduces noise from non-representative or irrelevant sample pairs.

In light of this limitation, the method employs Top-k similarity-based selection:

$$\begin{aligned} \mathcal{P}_{\text{top-}k} = \text{Top}_k\left(\text{similarity}(x_i, x_j)\right), \\ \forall\ (x_i, x_j) \in \mathcal{P}. \end{aligned} \tag{4}$$

Here, only the top-k most similar pairs are retained for training, reducing computational overhead while focusing on informative interactions. Despite this improvement, the diversity of effective samples remains fundamentally constrained by the original batch size B. In Section 3.4, we develop a strategy that empowers the model to mine global sample beyond the current batch.

3.2 Sequential Matryoshka Representation Learning

Applying the conclusions from Section 3.1 to the MRL training process, and take the parallel dimensionality reduction process $[D, D/2, D/4]$ as an example. The ratio of the average gradients for parameters $\mathbf{w}_i (i \in [0, D/4])$ and $\mathbf{w}_j (j \in [D/4, D/2])$ is as follows:

$$\overline{\text{grad}_i} : \overline{\text{grad}_j} = \left(\frac{\partial \mathcal{L}^{D}}{\partial \mathbf{w}_i} + \frac{\partial \mathcal{L}^{D/2}}{\partial \mathbf{w}_i} + \frac{\partial \mathcal{L}^{D/4}}{\partial \mathbf{w}_i} \right) : \left(\frac{\partial \mathcal{L}^{D}}{\partial \mathbf{w}_i} + \frac{\partial \mathcal{L}^{D/2}}{\partial \mathbf{w}_i} \right) \approx 1 + \frac{\delta (D/2)^2}{\delta (D/4)^2}. \tag{5}$$

As shown in Equation 10, the average gradient magnitude of parameter $\mathbf{w}_i$ can be approximated as $1 + \frac{\delta(D/2)^2}{\delta(D/4)^2}$ times that of parameter $\mathbf{w}_j$, primarily due to the influence of the lower-dimensional loss function $\mathcal{L}^{D/4}$. To resolve this issue, we propose Sequential Matryoshka Representation Learning (SMRL), which substitutes the original parallel compression of embeddings with a sequential approach, as illustrated in the Figure 2. Assuming a dimensionality reduction trajectory of $[D, D/2, D/4, \ldots, D/2^n]$. In each iteration, only the immediate transition (e.g., $D/2^{n-1} \rightarrow D/2^n$) is trained, avoiding the inclusion of lower-dimensional losses that amplify gradients for low-dimensional parameters. By eliminating the above factor, the gradients of $\mathbf{w}_i (i \in [0, D/2^n])$ follow a consistent distribution with reduced variance, improving convergence speed and performance. Once the loss converges in the current iteration, the dimensionality reduction $D/2^{n-1} \rightarrow D/2^n$ is complete, and the process proceeds to the next stage $D/2^n \rightarrow D/2^{n+1}$, repeating the procedure until the target dimension is reached. Additionally, after convergence in one iteration, the optimal parameters for the current dimension are fixed to prevent subsequent reductions from degrading their performance. Notably, compared to MRL, the SMRL framework is more amenable to continued training. In scenarios where low-dimensional retrieval embeddings (e.g., D/8) or intermediate embeddings (e.g., D/3) are required, these can be obtained through further dimensionality reduction training based on the already preserved D/4 or D/2 parameters, eliminating the need for retraining from scratch as is typically required in MRL.

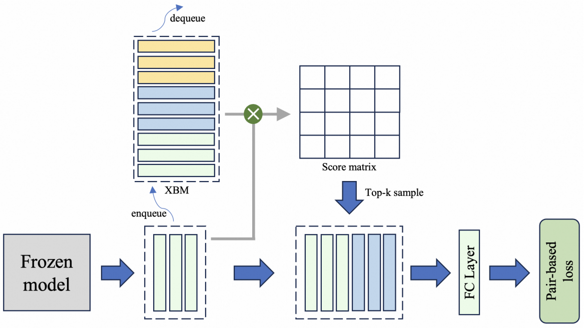


Figure 4: S-XBM maintains a queue during training to store historical features across batches. Rather than incorporating all stored features into the current batch, it selectively leverages hard samples that exhibit high similarity to the current batch samples.

3.3 Adaptive Dimension Selection Module

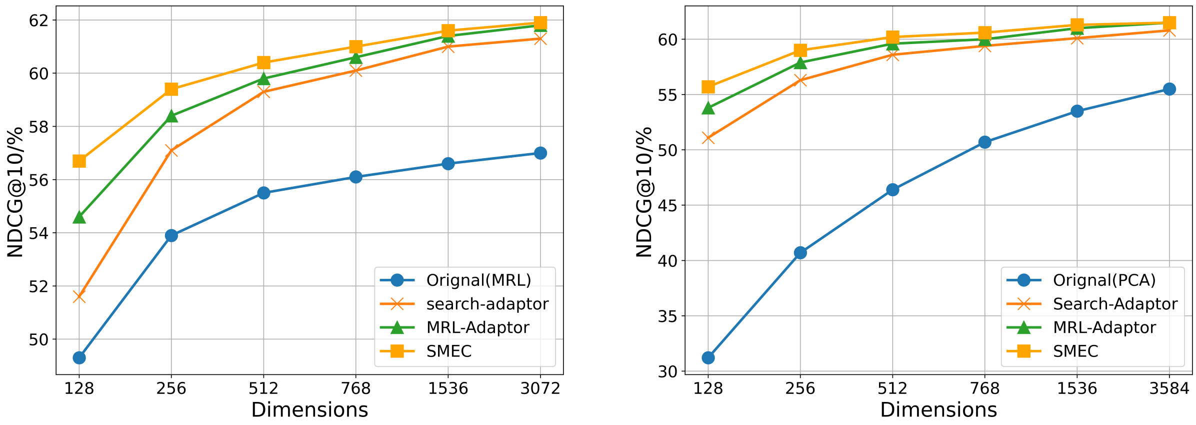
Since directly truncating dimensions to obtain low-dimensional representations in MRL inevitably leads to information degradation, we propose the Adaptive Dimension Selection (ADS) module to dynamically identify important dimensions during training. As illustrated in Figure 3, we introduce a set of learnable parameters that represent the importance of different dimensions in the original representation $\mathbf{Z}(\dim = D)$, and use these parameters to perform dimensional sampling, obtaining a reduced-dimension representation $\mathbf{Z}'(\dim = D/2)$. Since the sampling operation is non-differentiable, during the training phase, we utilize the Gumbel-Softmax (Jang et al., 2016) to approximate the importance of different dimensions. This is achieved by adding Gumbel-distributed noise $G \sim \text{Gumbel}(0, 1)$ to the logits parameters $\hat{\mathbf{z}}$ for each dimension, followed by applying the softmax function to the perturbed logits to approximate the one-hot vector representing dimension selection. Mathematically, this can be expressed as:

$$\mathbf{z} = \text{softmax}_{\tau}(\hat{\mathbf{z}} + G). \tag{6}$$

Importantly, the Gumbel approximation allows the softmax scores of dimension importance to be interpreted as the probability of selecting each dimension, rather than enforcing a deterministic selection of the top-k dimensions. This achieves a fully differentiable reparameterization, transforming the selection of embedding dimensions into an optimizable process.

3.4 Selectable Cross-Batch Memory

A natural teacher-student relationship inherently exists between the original embedding and its reduced-dimensional counterpart, making it feasible to improve the compressed embedding through unsupervised learning (Yoon et al., 2024). However, as discussed in Section 3.1, performing this process within a single batch suffers from sample noise and insufficient diversity. As illustrated in



(a) OpenAI text embeddings

(b) LLM2Vec

Figure 5: Experimental results on the BEIR dataset comparing two models: OpenAI's text-embedding-3-large (with 3072 dimensions) and LLM2Vec (with 3548 dimensions), the latter built upon the Qwen2-7B model. OpenAI text embeddings inherently contain multi-scale representations (enabled by MRL during pretraining), while LLM2Vec obtains its orignal low-dimensional representations via PCA.

Figure 4, we propose the Selectable Cross-Batch Memory (S-XBM) module, which constructs a first-in-first-out (FIFO) queue during training to store original embeddings across batches, with the aim of addressing this limitation. Unlike the original XBM (Wang et al., 2020), we introduce two task-specific improvements: (1) retrieving only the top-k most similar samples from the memory bank to construct new batches, and (2) deferring the trainable FC layer and only storing features generated by the frozen backbone, thereby avoiding feature drift. The unsupervised loss between original embedding emb and low-dimensional embedding $emb[:d]$ is as follows:

$$\mathcal{L}_{un-sup} = \sum_{i} \sum_{j \in \mathcal{N}_K(i)} |\text{Sim}(emb_i, emb_j) - \text{Sim}(emb_i[:d], emb_j[:d])| \quad (7)$$

where $\mathcal{N}_K(i)$ denotes the set of the top k most similar embeddings to emb_i within the S-XBM module.

4 Experiments

In this section, we compare our approach with state-of-the-art methods in the field of embedding dimensionality reduction.

4.1 Dataset Description

We evaluate the model's retrieval performance across diverse datasets: BEIR (Thakur et al., 2021) (text retrieval), Products-10K (Bai et al., 2020) (image retrieval), and Fashion-200K (Han et al., 2017) (cross-modal retrieval). BEIR is a comprehensive text retrieval benchmark consisting of 13 selected datasets from diverse domains. Products-10K contains approximately 10,000 products with over 150,000 images for large-scale product image retrieval. Fashion-200K includes over 200,000 fashion items with paired image-text data for cross-modal tasks.

4.2 Implementation Details

We use state-of-the-art models to extract the original embeddings for different datasets. Specifically, the BEIR dataset employs OpenAI text embeddings (ope) and LLM2Vec (BehnamGhader et al., 2024) for text representation; the Products-10K dataset utilizes LLM2CLIP (Huang et al., 2024) to obtain cross-modal embeddings; and the Fashion-200K dataset extracts image embeddings using the ViT-H(Dosovitskiy et al., 2020) model. All dimensionality reduction methods are performed based on these original representations. To align with other methods, SMEC also adopts rank loss (Yoon et al., 2023) as the supervised loss function, which is defined as follows:

$$\mathcal{L}_{rank} = \sum_{i} \sum_{j} \sum_{k} \sum_{m} I(y_{ij} > y_{ik})(y_{ij} - y_{ik}) \log(1 + \exp(s_{ik}[:m] - s_{ij}[:m])), \quad (8)$$

where $I(y_{ij} > y_{ik})$ is an indicator function that is equal to 1 if $y_{ij} > y_{ik}$ and 0 otherwise. $s_{ij}[:m]$ represents the cosine similarity between the query embedding $emb_i[:m]$ and the corpus embedding

$emb_j[:m]$. The total loss function is:

$$\mathcal{L}_{total} = \mathcal{L}_{rank} + \alpha \mathcal{L}_{un-sup}, \quad (9)$$

with α being hyper-parameters with fixed values as $\alpha = 1.0$. As SMEC involves multi-stage training, the training epochs of other methods are aligned with the total number of epochs costed by SMEC, and their best performance is reported.

4.3 Results

In this subsection, the results on the BEIR, Fashion-200K, and Products-10K datasets are given. Retrieval performance is evaluated using the normalized discounted cumulative gain at rank 10 (nDCG@10)(Kalervo et al., 2002) metric.

BEIR. As shown in Figure 5, we compare the performance of SMEC and other state-of-the-art methods on two types of models: the API-based OpenAI text embedding and the open-source LLM2vec, across various compressed dimensions. Significantly, SMEC exhibits the strongest performance retention, particularly at lower compression ratios. For example, when compressed to 128 dimensions, SMEC improves the performance of the OpenAI and LLM2vec models by 1.9 and 1.1 points respectively, compared to the best-performing Matryoshka-Adaptor.

Products-10K. Images naturally contain denser features than text (O Pinheiro et al., 2020). As shown in Figure 8a of Appendix C, SMEC surpasses other dimensionality reduction methods in image retrieval tasks, highlighting the effectiveness of the ADS module in mitigating information degradation during dimension pruning.

Fashion-200K. Unlike unimodal datasets, Fashion-200K involves cross-modal queries and documents, such as image-to-text and text-to-image retrieval. As illustrated in the Figure 8b and 8c of Appendix C, SMEC achieves superior performance in both directions, demonstrating strong robustness in multimodal scenarios.

5 Discussions

5.1 The influence of gradient variance

To validate the impact of gradient variance on convergence speed and model performance (as discussed in Section 3.2), we conducted comparative experiments between SMRL and MRL using the MiniLM model on the BEIR dataset. As shown in Figure 6a, MRL consistently exhibits significantly higher gradient variance than SMRL throughout training. Consequently, the training loss of MRL continues to decline beyond the 20th epoch, whereas SMRL's loss starts to converge at the 15th epoch. A similar trend is observed in subfigure 6c, where SMRL enters the improvement phase earlier and converges to superior performance.

5.2 Gradient variance of different loss functions

Section 5.1 demonstrates that MRL exhibits higher gradient variance compared to SMRL when rank loss is employed as the loss function, thereby corroborating the findings presented in Section 3.2. To enhance the validation, we conducted additional experiments on the BEIR dataset using rank loss, MSE loss and cross-entropy (CE) loss under identical settings. The results depicted in Figure 7 reveal a consistent pattern across both loss functions, validating the robustness of our conclusions.

5.3 Ablation studies

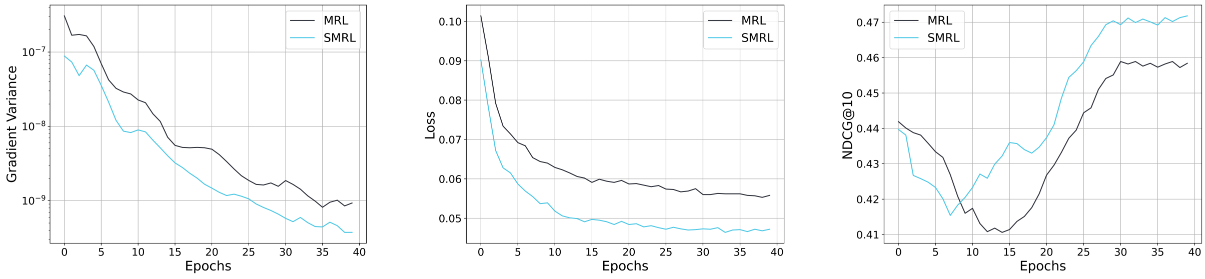
To evaluate the contribution of each component in SMEC to the overall performance, we conduct ablation studies using MRL as the baseline. Different modules are incrementally added on top of MRL, as detailed in table 1. When examined individually, the SMRL strategy achieves the most significant performance gain, suggesting that its reduced gradient variance contributes positively to model performance. In addition, both the ADS module and the S-XBM module also provide notable improvements. The combination of all three components improves the performance of the 128-dimensional embedding by 3.1 points.

Method	64	128	256	512
MRL (Baseline)	0.3726	0.4534	0.4802	0.5207
w/ SMRL	0.3808	0.4621	0.4895	0.5283
w/ ADS	0.3765	0.4583	0.4863	0.5254
w/ S-XBM	0.3778	0.4583	0.4853	0.5256
SMEC (Ours)	**0.4053**	**0.4848**	**0.5002**	**0.5459**

Table 1: Ablation studies of SMEC on 8 BEIR datasets with MRL as the baseline.

5.4 The contribution of ADS in preserving key information

The selection of important parameters in neural networks is a well-established research area, with numerous studies demonstrating that network parameters are often redundant. As a result, Parameter Pruning have been widely adopted for model



(a) Gradient Variance

(b) Validation loss

(c) Retrieval performance

Figure 6: Analysis of metrics during the training process. (a) shows the gradient variance curve (with the vertical axis in a logarithmic scale), (b) presents the loss curve on the validation set, and (c) illustrates the performance variations on the test set. As training progresses, the gradient variances of both MRL and SMRL decrease; however, the gradient variance of MRL remains several times higher than that of SMRL. Consequently, the loss curve of SMRL converges more quickly to a lower value, and the compressed embedding demonstrates better retrieval performance.

compression. We consider ADS (or more generally, the MEC family of methods), although it focuses on dimension selection within embeddings, to be fundamentally implemented through network parameter selection. Therefore, ADS can be regarded as a form of pruning method with theoretical feasibility.

To fully demonstrate the effectiveness of ADS, we evaluate both the dimension selection strategies of ADS and MRL using WARE(Yu et al., 2018) (Weighted Average Reconstruction Error), a commonly used metric in the pruning area for assessing parameter importance. The WARE is defined as follows:

$$\text{WARE} = \frac{1}{M}\sum_{m=1}^{M}\frac{|\hat{y}_m - y_m|}{|y_m|} \tag{10}$$

,where M denotes the number of samples; $\hat{y}_m$ and y_m represent the model's score (which can be interpreted as the similarity between embedding pairs) for the m-th sample before and after dimension pruning, respectively. The core idea of WARE is to quantify the change in the model's output induced by removing a specific dimension; a larger change indicates higher importance of that dimension.

We randomly sampled 10,000 instances from multiple sub-datasets of BEIR. For the LLM2VEC embeddings (3072dim), we computed the WARE for each dimension. Then, we used both ADS and MRL to generate low-dimensional embeddings of 1536, 768, and 256 dimensions, respectively. For each method and compression level, we calculated the achievement rate, which is defined as the proportion of selected dimensions that appear in the top-N most important dimensions according to the WARE-based ranking.

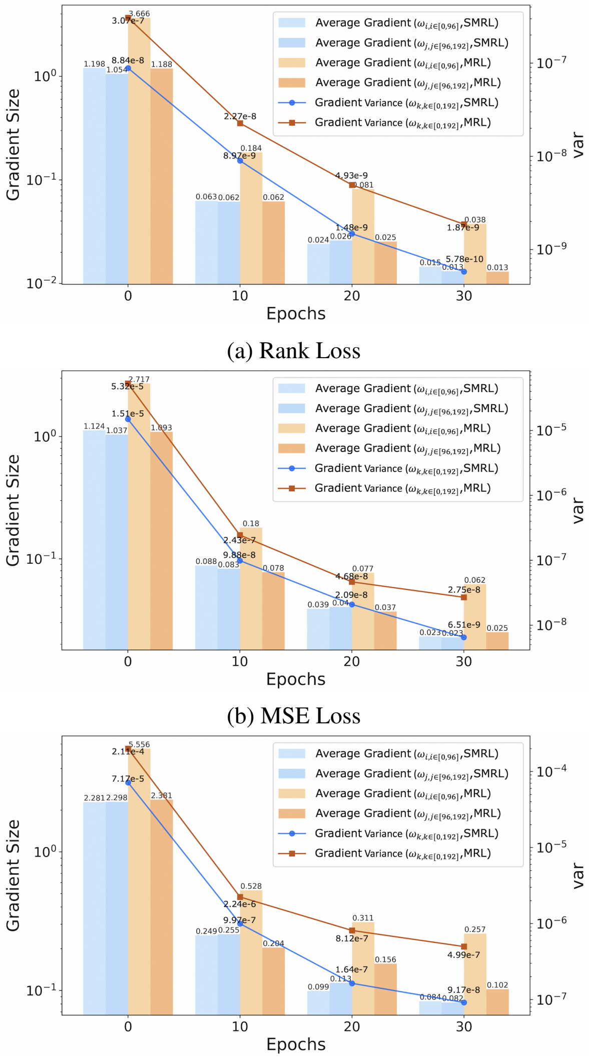
Dimension	ADS (Dimension Selection)	MRL (Dimension Truncation)
1536	94.3%	50.3%
768	90.1%	32.8%
256	83.6%	17.4%

Table 2: Achievement Rate of Important Dimension Selection at Different Dimension Levels.

The results in table 2 show that the achievement rate of MRL is roughly linear with the compression ratio, indicating that the importance of dimensions has no strong correlation with their positions. The achievement rate of ADS also decreases as the number of retained dimensions reduces, which is due to the increased difficulty of selecting the top-N most important dimensions under higher compression ratios. However, even when compressed by a factor of 6, ADS still selects over 80 of the most important dimensions. This explains why, as seen in Figure 5, SMEC demonstrates stronger performance at lower dimensions.

5.5 Memory size of S-XBM

In this subsection, we explore how the memory size of S-XBM module affects training speed and model performance. Theoretically, as the memory size increases, it is easier for the S-XBM module to mine more hard samples, thereby improving model performance. However, an excessively large memory size may increase the retrieval time for top-k samples, which could negatively affect training efficiency. To prove this observation experimentally, we train the SMEC framework with varying memory sizes (e.g., 1000, 2000, 5000, 10000,



(a) Rank Loss

(b) MSE Loss

(c) CE Loss

Figure 7: Gradient statistics with Rank, MSE and CE loss (with the vertical axis in a logarithmic scale): average gradient magnitudes of parameters in the ranges $[0, 96]$ and $[96, 192]$, as well as the gradient variance over all parameters in the range $[0, 192]$, during training.

and 15000), as illustrated in the table 3. The results demonstrate a clear trade-off between training speed and model performance. We select a memory size of 5000 as our final choice to strike a balance between them.

6 Conclusions

Although high-dimensional embeddings from large language models (LLMs) capture rich semantic features, their practical use is often limited by computational efficiency and storage constraints. To mitigate these limitations, Sequential Matryoshka Embedding Compression (SMEC) framework is proposed in this paper to achieve efficient embedding compression. Our proposed SMEC framework contains Sequential Matryoshka Representation Learning(SMRL) module, adaptive dimension selection (ADS) module and Selectable Cross-batch Memory (S-XBM) module. The SMRL module is designed to mitigate gradient variance during training. The ADS module is utilized to minimize information degradation during feature compression. And the S-XBM is utilized to enhance unsupervised learning between high- and low-dimensional embeddings. Compared to existing approaches, our approaches preserve higher performance at the same compression rate.

Memory Size	1000	2000	5000	10000	15000
Forward Time/s ↓	0.06	0.08	0.11	0.15	0.21
NDCG@10 ↑	0.4631	0.4652	0.4675	0.4682	0.4689

Table 3: Trade-off analysis of training speed and model performance under different memory size of S-XBM.

Limitations

The SMEC framework introduces only a small number of additional parameters on top of a pre-trained model and is trained using labeled data from a specific domain, along with mined hard samples, with the aim of reducing the dimensionality of the original embeddings. However, this design and objective limit its generalizability and applicability to broader scenarios. Future work could explore extending the SMEC approach to full-parameter training of representation models, enabling them to directly generate embeddings of multiple dimensions. Additionally, the feasibility of training the model on diverse datasets is also worth investigating.

References

Openai text embedding. https://platform.openai.com/docs/guides/embeddings/embeddings.

Yalong Bai, Yuxiang Chen, Wei Yu, Linfang Wang, and Wei Zhang. 2020. Products-10k: A large-scale product recognition dataset.

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. Llm2vec: Large language models are secretly powerful text encoders. *arXiv preprint arXiv:2404.05961*.

K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. 1999. When is nearest neighbor meaningful. *Springer*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Mu Cai, Jianwei Yang, Jianfeng Gao, and Yong Jae Lee. 2024. Matryoshka multimodal models. In *Workshop on Video-Language Models@ NeurIPS 2024*.

Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, and 1 others. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Jiatao Gu, Shuangfei Zhai, Yizhe Zhang, Joshua M Susskind, and Navdeep Jaitly. 2023. Matryoshka diffusion models. In *The Twelfth International Conference on Learning Representations*.

Xintong Han, Zuxuan Wu, Phoenix X. Huang, Xiao Zhang, and Larry S. Davis. 2017. Automatic spatially-aware fashion concept discovery. *IEEE*.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738.

Wenbo Hu, Zi-Yi Dou, Liunian Li, Amita Kamath, Nanyun Peng, and Kai-Wei Chang. 2024. Matryoshka query transformer for large vision-language models. *Advances in Neural Information Processing Systems*, 37:50168–50188.

Weiquan Huang, Aoqi Wu, Yifan Yang, Xufang Luo, Yuqing Yang, Liang Hu, Qi Dai, Xiyang Dai, Dongdong Chen, Chong Luo, and Lili Qiu. 2024. Llm2clip: Powerful language model unlock richer visual representation. *Preprint*, arXiv:2411.04997.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus.

Ian T. Jolliffe and Jorge Cadima. 2016. Principal component analysis: a review and recent developments. *Philos Trans A Math Phys Eng*, 374(2065):20150202.

Järvelin Kalervo, Jaana Kekäläinen, and Almahairi. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446.

Diederik P Kingma, Max Welling, and 1 others. 2013. Auto-encoding variational bayes.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and 1 others. 2022. Matryoshka representation learning. *Advances in Neural Information Processing Systems*, 35:30233–30249.

Daniel Lee and H Sebastian Seung. 2000. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, 13.

Yixiao Li, Yifan Yu, Qingru Zhang, Chen Liang, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2023. Losparse: Structured compression of large language models based on low-rank and sparse approximation. In *International Conference on Machine Learning*, pages 20336–20350. PMLR.

Geoffrey J Mclachlan. Discriminant analysis and statistical pattern recognition. *Wiley-Interscience,*.

Pedro O O Pinheiro, Amjad Almahairi, Ryan Benmalek, Florian Golemo, and Aaron C Courville. 2020. Unsupervised learning of dense visual representations. *Advances in neural information processing systems*, 33:4489–4500.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Stephen E. Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. *ACM*.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding.

Xun Wang, Haozhi Zhang, Weilin Huang, and Matthew R Scott. 2020. Cross-batch memory for embedding learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6388–6397.

Jinsung Yoon, Sercan O Arik, Yanfei Chen, and Tomas Pfister. 2023. Search-adaptor: Embedding customization for information retrieval. *arXiv preprint arXiv:2310.08750*.

Jinsung Yoon, Rajarishi Sinha, Sercan O Arik, and Tomas Pfister. 2024. Matryoshka-adaptor: Unsupervised and supervised tuning for smaller embedding dimensions. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10318–10336, Miami, Florida, USA. Association for Computational Linguistics.

Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. 2018. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9194–9203.

Yury, A, Malkov, Dmitry, A, and Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis Machine Intelligence*.

A Derivation of the Gradient Fluctuation

To formalize this issue, we analyze the Mean Squared Error (MSE) loss as a representative case. Let $\mathbf{x}_1 = [x_1, x_2, \ldots, x_n]^\top \in R^n$ and $\mathbf{x}_2 = [y_1, y_2, \ldots, y_n]^\top \in R^n$ denote two input feature vectors. The final FC layer employs a matrix $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n]^\top \in R^{m\times n}$ to generate scalar outputs $\mathbf{y}_1 = \mathbf{W}\mathbf{x}_1 \in R^m$ and $\mathbf{y}_2 = \mathbf{W}\mathbf{x}_2 \in R^m$. The MSE loss at dimension d is defined as:

$$\mathcal{L}^d = \left[\mathcal{Y}_{label} - sim(\mathbf{y}_1^d, \mathbf{y}_2^d)\right]^2, \tag{11}$$

where $\mathcal{Y}_{label}$ denotes the binary classification label for pairs (0 or 1), and $sim(\cdot)$ represents the normalized similarity of the learned representations.

According to the chain rule, the partial derivative of $\mathcal{L}^d$ with respect to the i-th dimension parameter of the FC layer is derived as:

$$\frac{\partial \mathcal{L}^d}{\partial \mathbf{w}_i} = \frac{\partial \mathcal{L}^d}{\partial \left[\mathbf{y}_1^d\right]_i} \cdot \frac{\partial \left[\mathbf{y}_1^d\right]_i}{\partial \mathbf{w}_i} + \frac{\partial \mathcal{L}^d}{\partial \left[\mathbf{y}_2^d\right]_i} \cdot \frac{\partial \left[\mathbf{y}_2^d\right]_i}{\partial \mathbf{w}_i}. \tag{12}$$

Utilizing cosine similarity (clamp to $[0, 1]$) as the similarity function $sim(\cdot)$, the equation 11 can be rewritten as:

$$\mathcal{L}^d = \left[\mathcal{Y}_{label} - \frac{\mathbf{y}_1^{d\top} \mathbf{y}_2^d}{\|\mathbf{y}_1^d\| \|\mathbf{y}_2^d\|}\right]^2. \tag{13}$$

Let $\|\mathbf{y}_1^d\| = A$, $\|\mathbf{y}_2^d\| = B$, $\mathbf{y}_1^{d\top} \mathbf{y}_2^d = C$ and $s = \frac{C}{AB}$. The partial derivatives of the $\mathcal{L}^d$ with respect to $\left[\mathbf{y}_1^d\right]_i$ and $\left[\mathbf{y}_2^d\right]_i$ are given as follows:

$$\frac{\partial \mathcal{L}^d}{\partial \left[\mathbf{y}_1^d\right]_i} = 2\left(s - \mathcal{Y}_{label}\right)\left(\frac{\left[\mathbf{y}_2^d\right]_i}{AB} - \frac{s}{A^2}\left[\mathbf{y}_1^d\right]_i\right), \tag{14}$$

$$\frac{\partial \mathcal{L}^d}{\partial \left[\mathbf{y}_2^d\right]_i} = 2\left(s - \mathcal{Y}_{label}\right)\left(\frac{\left[\mathbf{y}_1^d\right]_i}{AB} - \frac{s}{B^2}\left[\mathbf{y}_2^d\right]_i\right). \tag{15}$$

Substituting $\left[\mathbf{y}_1^d\right]_i = \mathbf{w}_i\mathbf{x}_1$ and $\left[\mathbf{y}_2^d\right]_i = \mathbf{w}_i\mathbf{x}_2$, the partial derivatives of the $\left[\mathbf{y}_1^d\right]_i$ and $\left[\mathbf{y}_2^d\right]_i$ with respect to $\mathbf{w}_i$ are given as follows:

$$\frac{\partial \left[\mathbf{y}_1^d\right]_i}{\partial \mathbf{w}_i} = \mathbf{x}_1, \frac{\partial \left[\mathbf{y}_2^d\right]_i}{\partial \mathbf{w}_i} = \mathbf{x}_2. \tag{16}$$

Based on the above equations, the partial derivative of $\mathcal{L}^d$ with respect to $\mathbf{w}_i$ is derived as:

$$\frac{\partial \mathcal{L}^d}{\partial \mathbf{w}_i} = 2\left(s - \mathcal{Y}_{label}\right)\left[\left(\frac{\left[\mathbf{y}_2^d\right]_i}{AB} - \frac{s}{A^2}\left[\mathbf{y}_1^d\right]_i\right)\mathbf{x}_1 + \left(\frac{\left[\mathbf{y}_1^d\right]_i}{AB} - \frac{s}{B^2}\left[\mathbf{y}_2^d\right]_i\right)\mathbf{x}_2.\right] \tag{17}$$

Assume that A and B can be approximated by $\delta(d)\cdot a$ and $\delta(d)\cdot b$, respectively. Under this approximation, $\delta(d)$ can be used to fit the relationship between the magnitude of vector $\mathbf{x}$ or $\mathbf{y}$ and the variation of d (It is evident that this is a positive correlation). Therefore, Equation 17 can be approximated by the following expression:

$$\frac{\partial \mathcal{L}^d}{\partial \mathbf{w}_i} = 2\left(s - \mathcal{Y}_{label}\right)\frac{1}{\delta(d)^2}\left[\left(\frac{\left[\mathbf{y}_2^d\right]_i}{ab} - \frac{s}{a^2}\left[\mathbf{y}_1^d\right]_i\right)\mathbf{x}_1 + \left(\frac{\left[\mathbf{y}_1^d\right]_i}{ab} - \frac{s}{b^2}\left[\mathbf{y}_2^d\right]_i\right)\mathbf{x}_2\right]. \tag{18}$$

In equation 18, a, b, $\left[\mathbf{y}_1^d\right]_i$, $\left[\mathbf{y}_2^d\right]_i$ are constants, $\mathbf{x}_1$ and $\mathbf{x}_2$ are constant vectors, while s and $\mathcal{Y}_{label}$ are invariant with respect to the index d. Therefore, we can conclude the following:

$$\frac{\partial \mathcal{L}^d}{\partial \mathbf{w}_i} \propto \frac{1}{\delta(d)^2}. \tag{19}$$

In theory, this rule can also be extended to other pair-wise similarity-based functions, such as rank loss, which is experimentally verified in Section 5.2.

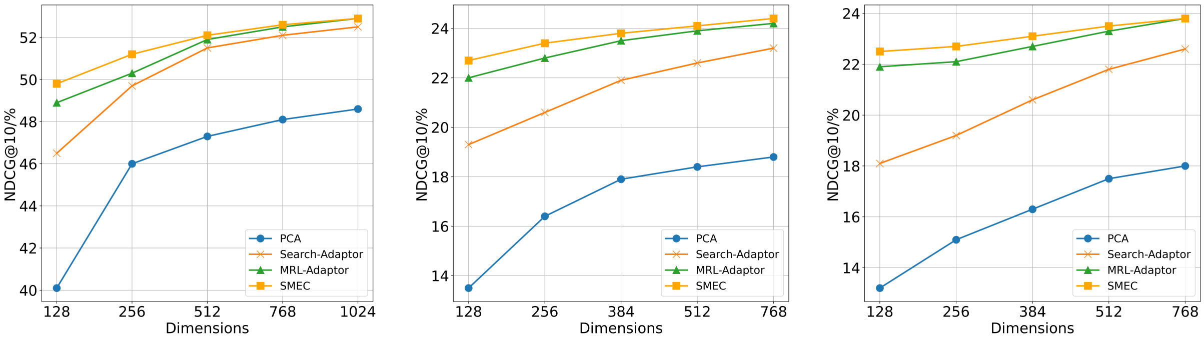
B Results on BEIR Sub-datasets.

We compare the performance of different compression methods on several representative sub-datasets of BEIR, and the results are shown in Table 4.

	NDCG@10					
Model	**128**	**256**	**512**	**768**	**1536**	**3072**
Sub-dataset–Scifact						
LLM2Vec	-	-	-	-	-	0.787
w/ Search-Adaptor	0.806	0.845	0.864	0.879	0.886	0.884
w/ MRL-Adaptor	0.826	0.861	0.876	0.880	0.886	0.887
w/ SMEC (ours)	0.841	0.874	0.879	0.882	0.885	0.886
Sub-dataset–FiQA						
LLM2Vec	-	-	-	-	-	0.498
w/ Search-Adaptor	0.475	0.505	0.529	0.540	0.545	0.550
w/ MRL-Adaptor	0.496	0.523	0.534	0.543	0.547	0.550
w/ SMEC (ours)	0.521	0.533	0.540	0.546	0.549	0.551
Sub-dataset–Quora						
LLM2Vec	-	-	-	-	-	0.775
w/ Search-Adaptor	0.771	0.805	0.830	0.845	0.861	0.864
w/ MRL-Adaptor	0.784	0.812	0.834	0.847	0.862	0.863
w/ SMEC (ours)	0.794	0.818	0.839	0.850	0.862	0.865
Sub-dataset–NFCorpus						
LLM2Vec	-	-	-	-	-	0.389
w/ Search-Adaptor	0.345	0.375	0.396	0.412	0.425	0.426
w/ MRL-Adaptor	0.364	0.384	0.403	0.419	0.426	0.427
w/ SMEC (ours)	0.389	0.402	0.418	0.426	0.430	0.431
Sub-dataset–SciDocs						
LLM2Vec	-	-	-	-	-	0.232
w/ Search-Adaptor	0.204	0.225	0.245	0.250	0.258	0.263
w/ MRL-Adaptor	0.220	0.240	0.250	0.255	0.262	0.265
w/ SMEC (ours)	0.239	0.246	0.251	0.255	0.261	0.264

Table 4: Comparison of retrieval performance on 5 BEIR sub-datasets.

C Experimental results on Products-10K and Fashion-200k.



(a) Image retrieval

(b) Text-to-Image retrieval

(c) Image-to-Text retrieval

Figure 8: Experimental results on image and multimodal datasets. (a) presents the results on the Products-10K dataset using an image representation model based on ViT-H (with 1024 dimensions). (b) and (c) show the results on the Fashion-200K dataset for text-to-image and image-to-text retrieval tasks, respectively, using the LLM2CLIP model (with 768 dimensions, base on ViT-L/14 and Llama-3.2-1B).